



1

2 **Web Services Security**
3 **Rights Expression Language (REL)**
4 **Token Profile 1.1**

5 **Committee Draft: 14 June 2005**

6 **Document Location:**

7 <http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.1.pdf>

8 **Errata Location:**

9 <http://www.oasis-open.org/committees/wss>

10 **Technical Committee:**

11 Web Services Security (WSS)

12 **Chairs:**

13 Kelvin Lawrence, IBM

14 Chris Kaler, Microsoft

15 **Editors:**

16 Thomas DeMartini, ContentGuard, Inc.

17 Anthony Nadalin, IBM

18 Chris Kaler, Microsoft

19 Ronald Monzillo, Sun

20 Phillip Hallam-Baker, Verisign

21 **Abstract:**

22 This document describes how to use ISO/IEC 21000-5 Rights Expressions with the Web
23 Services Security (WSS) specification.

24 **Status:**

25 The status of this document is Committee Draft. Please send comments to the editors.

26 If you are on the wss@lists.oasis-open.org list for committee members, send comments
27 there. If you are not on that list, subscribe to the wss-comment@lists.oasis-open.org list

28 and send comments there. To subscribe, send an email message to wss-comment-
29 request@lists.oasis-open.org with the word "subscribe" as the body of the message.
30 For patent disclosure information that may be essential to the implementation of this
31 specification, and any offers of licensing terms, refer to the Intellectual Property Rights
32 section of the OASIS Web Services Security Technical Committee (WSS TC) web page
33 at <http://www.oasis-open.org/committees/wss/ipr.php>. General OASIS IPR information
34 can be found at <http://www.oasis-open.org/who/intellectualproperty.shtml>.

Notices

36 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
37 that might be claimed to pertain to the implementation or use of the technology described in this
38 document or the extent to which any license under such rights might or might not be available;
39 neither does it represent that it has made any effort to identify any such rights. Information on
40 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
41 website. Copies of claims of rights made available for publication and any assurances of licenses
42 to be made available, or the result of an attempt made to obtain a general license or permission
43 for the use of such proprietary rights by implementors or users of this specification, can be
44 obtained from the OASIS Executive Director.

45 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
46 applications, or other proprietary rights which may cover technology that may be required to
47 implement this specification. Please address the information to the OASIS Executive Director.

48 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
49 2002-2005. All Rights Reserved.

50 This document and translations of it may be copied and furnished to others, and derivative works
51 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
52 published and distributed, in whole or in part, without restriction of any kind, provided that the
53 above copyright notice and this paragraph are included on all such copies and derivative works.
54 However, this document itself does not be modified in any way, such as by removing the
55 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
56 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
57 Property Rights document must be followed, or as required to translate it into languages other
58 than English.

59 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
60 successors or assigns.

61 This document and the information contained herein is provided on an "AS IS" basis and OASIS
62 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
63 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
64 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
65 PARTICULAR PURPOSE.

Table of Contents

67	1	Introduction (Informative).....	5
68	2	Notations and Terminology (Normative).....	6
69	2.1	Notational Conventions.....	6
70	2.2	Namespaces.....	6
71	2.3	Terminology.....	7
72	3	Usage (Normative).....	8
73	3.1	Token Types.....	8
74	3.2	Processing Model.....	8
75	3.3	Attaching Security Tokens.....	8
76	3.4	Identifying and Referencing Security Tokens.....	8
77	3.5	Authentication.....	11
78	3.5.1	<r:keyHolder> Principal.....	12
79	3.6	Confidentiality.....	14
80	3.6.1	<r:keyHolder> Principal.....	15
81	3.7	Error Codes.....	16
82	4	Types of Licenses (Informative).....	17
83	4.1	Attribute Licenses.....	17
84	4.2	Sender Authorization.....	18
85	4.3	Issuer Authorization.....	18
86	5	Threat Model and Countermeasures (Informative).....	21
87	5.1	Eavesdropping.....	21
88	5.2	Replay.....	21
89	5.3	Message Insertion.....	22
90	5.4	Message Deletion.....	22
91	5.5	Message Modification.....	22
92	5.6	Man-in-the-Middle.....	22
93	6	References.....	23
94		Appendix A: Acknowledgements.....	24
95		Appendix B: Revision History.....	26
96			

97

1 Introduction (Informative)

98 The Web Services Security: SOAP Message Security [WS-Security] specification proposes a
99 standard set of SOAP extensions that can be used when building secure Web services to
100 implement message level integrity and confidentiality. This specification describes the use of
101 ISO/IEC 21000-5 Rights Expressions with respect to the WS-Security specification.

102 2 Notations and Terminology (Normative)

103 This section specifies the notations, namespaces, and terminology used in this specification.

104 2.1 Notational Conventions

105 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
106 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be
107 interpreted as described in [KEYWORDS].

108 Namespace URIs (of the general form "some-URI") represent some application-dependent or
109 context-dependent URI as defined in [URI].

110 This specification is designed to work with the general SOAP message structure and message
111 processing model, and should be applicable to any version of SOAP. The current SOAP 1.2
112 namespace URI is used herein to provide detailed examples, but there is no intention to limit the
113 applicability of this specification to a single version of SOAP.

114 2.2 Namespaces

115 The following namespaces are used in this document:

116

Prefix	Namespace
S	http://www.w3.org/2001/12/soap-envelope
ds	http://www.w3.org/2000/09/xmlsig#
xenc	http://www.w3.org/2001/04/xmlenc#
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsse11	http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-wssecurity-secext-1.1.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
r	urn:mpeg:mpeg21:2003:01-REL-R-NS

sx	urn:mpeg:mpeg21:2003:01-REL-SX-NS
----	-----------------------------------

117

Table 1 Namespace Prefixes

118

119 **2.3 Terminology**

120 This specification employs the terminology defined in the Web Services Security: SOAP Message
121 Security [WS-Security] Specification.

122 Defined below are the basic definitions for additional terminology used in this specification.

123 **License** – ISO/IEC 21000-5 Rights Expression

124 3 Usage (Normative)

125 This section describes the syntax and processing rules for the use of licenses with
126 the Web Services Security: Soap Message Security specification [WS-Security].

127 3.1 Token Types

128 When a URI value is used to indicate a license according to this profile, its value MUST be
129 <http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf#license>.

130 3.2 Processing Model

131 The processing model for WS-Security with licenses is no different from that of WS-Security with
132 other token formats as described in Web Services Security: SOAP Message Security [WS-
133 Security].

134 At the token level, a processor of licenses MUST conform to the required validation and
135 processing rules defined in ISO/IEC 21000-5 [REL].

136 3.3 Attaching Security Tokens

137 Licenses are attached to SOAP messages using WS-Security by placing the license
138 element inside the `<wsse:Security>` header. The following example illustrates a
139 SOAP message with a license.

```
140 <S:Envelope xmlns:S="...">  
141   <S:Header>  
142     <wsse:Security xmlns:wsse="...">  
143       <r:license xmlns:r="...">  
144         ...  
145       </r:license>  
146       ...  
147     </wsse:Security>  
148   </S:Header>  
149   <S:Body>  
150     ...  
151   </S:Body>  
152 </S:Envelope>
```

153 3.4 Identifying and Referencing Security Tokens

154 The Web Services Security: SOAP Message Security [WS-Security] specification defines the
155 *wsu:id* attribute as the common mechanism for identifying security tokens (the specification
156 describes the reasons for this). Licenses have an additional identification mechanism available:
157 their *licenseId* attribute, the value of which is a URI. The following example shows a license that
158 uses both mechanisms:

159
160
161
162
163

```
<r:license xmlns:r="..." xmlns:wssu="..."
  licenseId="urn:foo:SecurityToken:ef375268"
  wssu:Id="SecurityToken-ef375268">
  ...
</r:license>
```

164 Licenses can be referenced either according to their location or their licenseld. Location
165 references are dependent on location and can be either local or remote. Licenseld references
166 are not dependent on location.

167 Local location references are RECOMMENDED when they can be used. Remote location
168 references are OPTIONAL for cases where it is not feasible to transmit licenses with the SOAP
169 message. Licenseld references are OPTIONAL for cases where location is unknown or cannot
170 be indicated.

171 WS-Security specifies that tokens are referenced using the <wsse:SecurityTokenReference>
172 element.

173 Implementations compliant with this profile SHOULD set the
174 /wsse:SecurityTokenReference/wsse:Reference/@ValueType attribute to http://docs.oasis-
175 open.org/wss/oasis-wss-rel-token-profile-1.0.pdf#license when using
176 wsse:SecurityTokenReference to refer to a license by licenseld. This is OPTIONAL when
177 referring to a license by location.

178 The following table demonstrates the use of the <wsse:SecurityTokenReference> element to
179 refer to licenses.

By Location	Local	<pre><wsse:SecurityTokenReference> <wsse:Reference URI="#SecurityToken-ef375268" /> </wsse:SecurityTokenReference></pre>
	Remote	<pre><wsse:SecurityTokenReference> <wsse:Reference URI="http://www.foo.com/ef375268.xml" /> </wsse:SecurityTokenReference></pre>
By licenseld		<pre><wsse:SecurityTokenReference> <wsse:Reference URI="urn:foo:SecurityToken:ef375268" ValueType="http://docs.oasis- open.org/wss/oasis-wss-rel-token-profile- 1.0.pdf#license" /> </wsse:SecurityTokenReference></pre>

180

Table 2. <wsse:SecurityTokenReference>

181 The following example demonstrates how a <wsse:SecurityTokenReference> can be used to
182 indicate that the message parts specified inside the <ds:SignedInfo> element were signed using
183 a key from the license referenced by licenseld in the <ds:KeyInfo> element.

184
185

```
<S:Envelope xmlns:S="...">
  <S:Header>
```

```

186     <wsse:Security xmlns:wsse="...">
187       <r:license xmlns:r="..."
188 licenseId="urn:foo:SecurityToken:ef375268" xmlns:wsu="..."
189 wsu:Id="SecurityToken-ef375268">
190         ...
191       </r:license>
192       ...
193     <ds:Signature>
194       <ds:SignedInfo>
195         ...
196       </ds:SignedInfo>
197       <ds:SignatureValue>...</ds:SignatureValue>
198       <ds:KeyInfo>
199         <wsse:SecurityTokenReference>
200           <wsse:Reference
201             URI="#SecurityToken-ef375268"
202           />
203         </wsse:SecurityTokenReference>
204       </ds:KeyInfo>
205     </ds:Signature>
206   </wsse:Security>
207 </S:Header>
208 <S:Body>
209   ...
210 </S:Body>
211 </S:Envelope>

```

212 The following example shows a signature over a local license using a location reference to that
213 license. The example demonstrates how the integrity of an (unsigned) license can be preserved
214 by signing it in the <wsse:Security> header.

```

215 <S:Envelope xmlns:S="...">
216   <S:Header>
217     <wsse:Security xmlns:wsse="..."
218       <r:license xmlns:r="..." xmlns:wsu="..." wsu:Id="SecurityToken-
219 ef375268">
220       ...
221     </r:license>
222     ...
223     <wsse:SecurityTokenReference wsu:Id="Str1">
224       <wsse:Reference
225         URI="#SecurityToken-ef375268"
226       />
227     </wsse:SecurityTokenReference>
228     ...
229     <ds:Signature>
230       <ds:SignedInfo>
231         ...
232       <Reference URI="#Str1">
233         <Transforms>
234           <ds:Transform
235             Algorithm="http://schemas.xmlsoap.org/2003/06/STR-
236 Transform">
237             <ds:CanonicalizationMethod
238               Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
239 20010315"/>
240           </ds:Transform>

```

```

241         </ds:Transforms>
242         <ds:DigestMethod
243             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
244         />
245         <ds:DigestValue>...</ds:DigestValue>
246     </ds:Reference>
247 </ds:SignedInfo>
248 <ds:SignatureValue>...</ds:SignatureValue>
249 <ds:KeyInfo>...</ds:KeyInfo>
250 </ds:Signature>
251 </wsse:Security>
252 </S:Header>
253 <S:Body>
254     ...
255 </S:Body>
256 </S:Envelope>

```

257 Note: since licenses allow the use of the wsu:Id attribute, it is usually not necessary to use the
258 STR-Transform because the license can be referred to directly in the ds:SignedInfo as shown in
259 the following example:

```

260 <S:Envelope xmlns:S="...">
261   <S:Header>
262     <wsse:Security xmlns:wsse="...">
263       <r:license xmlns:r="..." xmlns:wsu="..." wsu:Id="SecurityToken-
264 ef375268">
265         ...
266       </r:license>
267       ...
268       <ds:Signature>
269         <ds:SignedInfo>
270           ...
271           <ds:Reference URI="#SecurityToken-ef375268">
272             <ds:DigestMethod
273               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
274             />
275             <ds:DigestValue>...</ds:DigestValue>
276           </ds:Reference>
277         </ds:SignedInfo>
278         <ds:SignatureValue>...</ds:SignatureValue>
279         <ds:KeyInfo>...</ds:KeyInfo>
280       </ds:Signature>
281     </wsse:Security>
282   </S:Header>
283   <S:Body>
284     ...
285   </S:Body>
286 </S:Envelope>

```

287 3.5 Authentication

288 The Web Services Security: SOAP Message Security [WS-Security] specification does not dictate
289 how claim confirmation must be performed. As well, the REL allows for multiple types of
290 confirmation. This profile of WS-Security REQUIRES that message senders and receivers
291 support claim confirmation for <r:keyHolder> principals. It is RECOMMENDED that an XML

292 Signature be used to establish the relationship between the message sender and the claims. This
293 is especially RECOMMENDED whenever the SOAP message exchange is conducted over an
294 unprotected transport.

295 The following table enumerates the mandatory principals to be supported by claim confirmation
296 and summarizes their associated processing models. It should be noted that this table is not all-
297 encompassing, and it is envisioned that future specifications may expand this table over time.

Principal	RECOMMENDED Processing Rules
<r:keyHolder>	The message sender adds (to the security header) an XML Signature that can be verified with the key information specified in the <r:keyHolder> of the referenced license.

298 **Table 3. Processing Rules for Claim Confirmation**

299 Note that the high-level processing model described in the following sections does not
300 differentiate between message author and message sender as would be necessary to guard
301 against replay attacks. The high-level processing model also does not take into account
302 requirements for authentication of receiver by sender or for message or token confidentiality.
303 These concerns must be addressed by means other than those described in the high-level
304 processing model. If confidentiality of the token in the message is important, then use the
305 approach defined by [WS-Security] to encrypt the token.

306 **3.5.1 <r:keyHolder> Principal**

307 The following sections describe the <r:keyHolder> method of establishing the correspondence
308 between a SOAP message sender and the claims within a license.

309 **Sender**

310 The message sender MUST include within the <wsse:Security> header element a <r:license>
311 containing at least one <r:grant> to an <r:keyHolder> identifying the key to be used to confirm the
312 claims. If the message sender includes an <r:license> containing more than one <r:grant> to an
313 <r:keyHolder>, then all of those <r:keyHolder> elements MUST be equal.

314 In order for the receiver to perform claim confirmation, the sender MUST demonstrate knowledge
315 of the confirmation key. The sender MAY accomplish this by using the confirmation key to sign
316 content from within the message and by including the resulting <ds:Signature> element in the
317 <wsse:Security> header element. <ds:Signature> elements produced for this purpose MUST
318 conform to the canonicalization and token inclusion rules defined in the core WS-Security
319 specification and this profile specification.

320 Licenses that contain at least one <r:grant> to an <r:keyHolder> SHOULD contain an <r:issuer>
321 with a <ds:Signature> element that identifies the license issuer to the relying party and protects
322 the integrity of the confirmation key established by the license issuer.

323 Receiver

324 If the receiver determines that the sender has demonstrated knowledge of a confirmation key as
325 specified in an <r:keyHolder>, then the claims (found in the licenses) pertaining to that
326 <r:keyHolder> MAY be attributed to the sender. If one of these claims is an identity and if the
327 conditions of that claim are satisfied, then any elements of the message whose integrity is
328 protected by the confirmation key MAY be considered to have been authored by that identity.

329 Example

330 The following example illustrates how a license security token having an <r:keyHolder> principal
331 can be used with a <ds:Signature> to establish that John Doe is requesting a stock report on
332 FOO.

```
333 <S:Envelope xmlns:S="...">
334   <S:Header>
335     <wsse:Security xmlns:wsse="...">
336       <r:license xmlns:r="..."
337 licenseId="urn:foo:SecurityToken:ef375268">
338         <r:grant>
339           <r:keyHolder>
340             <r:info>
341               <ds:KeyValue>...</ds:KeyValue>
342             </r:info>
343           </r:keyHolder>
344           <r:possessProperty/>
345           <sx:commonName xmlns:sx="...">John Doe</sx:commonName>
346         </r:grant>
347         <r:issuer>
348           <ds:Signature>...</ds:Signature>
349         </r:issuer>
350       </r:license>
351     </S:Header>
352   <ds:Signature>
353     <ds:SignedInfo>
354       ...
355       <ds:Reference URI="#MsgBody">
356         <ds:DigestMethod
357           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
358         />
359         <ds:DigestValue>...</ds:DigestValue>
360       </ds:Reference>
361     </ds:SignedInfo>
362     <ds:SignatureValue>...</ds:SignatureValue>
363     <ds:KeyInfo>
364       <wsse:SecurityTokenReference>
365         <wsse:Reference
366           URI="urn:foo:SecurityToken:ef375268"
367           ValueType="http://docs.oasis-open.org/wss/oasis-wss-rel-
368 token-profile-1.0.pdf#license"
369         />
370       </wsse:SecurityTokenReference>
371     </ds:KeyInfo>
372   </ds:Signature>
373 </S:Envelope>
```

374
375
376
377
378
379
380
381
382
383
384
385

```

    </ds:Signature>

    </wsse:Security>
  </S:Header>

  <S:Body @wsu:Id="MsgBody" xmlns:wsu="...">
    <ReportRequest>
      <TickerSymbol>FOO</TickerSymbol>
    </ReportRequest>
  </S:Body>
</S:Envelope>

```

386 **3.6 Confidentiality**

387 This section details how licenses may be used to protect the confidentiality of a SOAP message
 388 within WS-Security. The Web Services Security: SOAP Message Security [WS-Security]
 389 specification does not dictate how confidentiality must be performed. As well, the REL allows for
 390 multiple types of confidentiality. This profile of WS-Security REQUIRES that message senders
 391 and receivers support confidentiality for <r:keyHolder> principals. It is RECOMMENDED that
 392 XML Encryption be used to ensure confidentiality. This is especially RECOMMENDED whenever
 393 the SOAP message exchange is conducted over an unprotected transport.

394 The following table enumerates the mandatory principals to be supported for confidentiality and
 395 summarizes their associated processing models. It should be noted that this table is not all-
 396 encompassing, and it is envisioned that future specifications may expand this table over time.

Principal	RECOMMENDED Processing Rules
<r:keyHolder>	The message sender adds (to the security header) either 1) an <xenc:ReferenceList> that points to one or more <xenc:EncryptedData> elements that can be decrypted with a key which can be determined from information specified in the <r:keyHolder> of the referenced license or 2) an <xenc:EncryptedKey> that can be decrypted with a key determined from information specified in the <r:keyHolder> of the referenced license.

397 **Table 4. Processing Rules for Confidentiality**

398 Note that this section deals only with Confidentiality. Details of authentication of the sender by
 399 the receiver must be addressed by means other than those described in this section (see the
 400 previous section).

401 3.6.1 <r:keyHolder> Principal

402 The following sections describe the <r:keyHolder> method of establishing confidentiality using a
403 license.

404 **Sender**

405 The message sender MUST include within the <wsse:Security> header element a <r:license>
406 containing at least one <r:grant> to an <r:keyHolder> identifying the key used to encrypt some
407 data or key. If the message sender includes an <r:license> containing more than one <r:grant> to
408 an <r:keyHolder>, then all of those <r:keyHolder> elements MUST be equal.

409 In order for the receiver to know when to decrypt the data or key, the sender MUST indicate the
410 encryption in the message. The sender MAY accomplish this by placing an
411 <xenc:EncryptedData> or <xenc:EncryptedKey> in the appropriate place in the message and by
412 including the resulting <xenc:ReferenceList> or <xenc:EncryptedKey> element in the
413 <wsse:Security> header element. <xenc:ReferenceList> or <xenc:EncryptedKey> elements
414 produced for this purpose MUST conform to the rules defined in the core WS-Security
415 specification and this profile specification.

416 **Receiver**

417 If the receiver determines that he has knowledge of a decryption key as specified in an
418 <r:keyHolder>, then he MAY decrypt the associated data or key. In the case of decrypting a key,
419 he may then recursively decrypt any data or key that that key can decrypt.

420

421 **Example**

422 The following example illustrates how a license containing a <r:keyHolder> principal can be used
423 with XML encryption schema elements to protect the confidentiality of a message using a
424 separate encryption key given in the <xenc:EncryptedKey> in the security header.

425 In this example, the r:license element provides information about the recipient's RSA public key
426 (i.e., KeyValue in keyHolder) used to encrypt the symmetric key carried in the EncryptedKey
427 element. The recipient uses this information to determine the correct private key to use in
428 decrypting the symmetric key. The symmetric key is then used to decrypt the EncryptedData child
429 of the Body element.

430

```
431 <S:Envelope xmlns:S="...">  
432   <S:Header>  
433     <wsse:Security xmlns:wsse="...">  
434       <r:license xmlns:r="...">  
435         licenseId="urn:foo:SecurityToken:ef375268">  
436           <r:grant>  
437             <r:keyHolder>  
438               <r:info>  
439                 <ds:KeyValue>...</ds:KeyValue>  
440               </r:info>
```

```

441     </r:keyHolder>
442     <r:possessProperty/>
443     <sx:commonName xmlns:sx="...">SOME COMPANY</sx:commonName>
444 </r:grant>
445 <r:issuer>
446     <ds:Signature>...</ds:Signature>
447 </r:issuer>
448 </r:license>
449 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
450     <xenc:EncryptionMethod
451         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
452     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
453         <wsse:SecurityTokenReference>
454             <wsse:Reference URI="urn:foo:SecurityToken:ef375268"/>
455         </wsse:SecurityTokenReference>
456     </KeyInfo>
457     <xenc:CipherData>
458         <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
459     </xenc:CipherData>
460     <xenc:ReferenceList>
461         <xenc:DataReference URI="#enc"/>
462     </xenc:ReferenceList>
463 </xenc:EncryptedKey>
464 </wsse:Security>
465 </S:Header>
466 <S:Body wsu:Id="body"
467     xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
468     <xenc:EncryptedData Id="enc"
469         Type="http://www.w3.org/2001/04/xmlenc#Content"
470         xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
471         <xenc:EncryptionMethod
472             Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
473         <xenc:CipherData>
474             <xenc:CipherValue>d2s...GQ</xenc:CipherValue>
475         </xenc:CipherData>
476         </xenc:EncryptedData>
477     </S:Body>
478 </S:Envelope>

```

479 3.7 Error Codes

480 It is RECOMMENDED that the error codes defined in the Web Services Security:
481 SOAP Message Security [WS-Security] specification are used. However,
482 implementations MAY use custom errors, defined in private namespaces if they
483 desire. Care should be taken not to introduce security vulnerabilities in the errors
484 returned.

485

4 Types of Licenses (Informative)

486

4.1 Attribute Licenses

487

In addition to key information, licenses can carry information about attributes of those keys.

488

Examples of such information on a client are e-mail address or common name. A service's key,

489

on the other hand, might be associated with a DNS name and common name.

490

The following is an example client attribute license.

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

```

<r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
  <r:inventory>
    <r:keyHolder licensePartId="client">
      <r:info>
        <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
      </r:info>
    </r:keyHolder>
  </r:inventory>
  <r:grant>
    <r:keyHolder licensePartIdRef="client"/>
    <r:possessProperty/>
    <sx:commonName>John Doe</sx:commonName>
  </r:grant>
  <r:grant>
    <r:keyHolder licensePartIdRef="client"/>
    <r:possessProperty/>
    <sx:emailName>jd@foo.com</sx:emailName>
  </r:grant>
  <r:issuer>
    <ds:Signature>...</ds:Signature>
  </r:issuer>
</r:license>

```

513

The following is an example service attribute license.

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

```

<r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
  <r:inventory>
    <r:keyHolder licensePartId="service">
      <r:info>
        <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
      </r:info>
    </r:keyHolder>
  </r:inventory>
  <r:grant>
    <r:keyHolder licensePartIdRef="service"/>
    <r:possessProperty/>
    <sx:commonName>MyService Company</sx:commonName>
  </r:grant>
  <r:grant>
    <r:keyHolder licensePartIdRef="service"/>
    <r:possessProperty/>
    <sx:dnsName>www.myservice.com</sx:dnsName>
  </r:grant>
  <r:issuer>
    <ds:Signature>...</ds:Signature>
  </r:issuer>
</r:license>

```

536 Additional examples of and processing rules for the use of attribute licenses can be found in the
537 above sections on Authentication and Confidentiality.

538 4.2 Sender Authorization

539 Licenses may be used by a sender as proof of authorization to perform a certain action on a
540 particular resource. This WS-Security specification does not describe how authorization must be
541 performed. In the context of web services, a sender can send to a receiver an authorization
542 license in the security header as proof of authorization to call the sender. Typically, this
543 authorization license is signed by a trusted authority and conforms to the syntax pattern specified
544 below.

```
545 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">  
546   <r:grant>  
547     <r:keyHolder>  
548       <r:info>  
549         <ds:KeyValue>FDFEWEFF...</ds:KeyValue>  
550       </r:info>  
551     </r:keyHolder>  
552     <sx:rightUri definition='...'/>  
553     <x:someResource/>  
554     <x:someCondition/>  
555   </r:grant>  
556   <r:issuer>  
557     <ds:Signature>...</ds:Signature>  
558   </r:issuer>  
559 </r:license>
```

560 The above license contains an authorization grant authorizing the keyholder (sender's public
561 key), the right to exercise the right identified in the <sx:rightUri> element. The resource in the
562 license typically corresponds to the semantics of the URI given in the definition attribute of the
563 <sx:rightUri> element. The entire license along with the <ds:Signature> element in the <r:issuer>
564 certifies the fact that the principal (<keyholder>) is granted the authorization to exercise the right
565 in the <sx:rightUri> element over the specified resource. The integrity of the license is usually
566 protected with a digital signature contained within the <ds:Signature>.

567 4.3 Issuer Authorization

568 To enunciate that a particular issuer is allowed to issue particular types of licenses, one can use
569 the kind of license described here. Issuer authorization licenses can accompany other licenses in
570 the security header such as those used for authentication, sender authorization, or other issuer
571 authorizations. These issuer authorization licenses might help complete the authorization proof
572 that is required for authorizing or authenticating a particular sender.

573

574 The following license is an example issuer authorization license for authorizing an issuer to issue
575 a simple attribute license.

```
576 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">  
577   <r:grant>  
578     <r:forAll varName='K' />  
579     <r:forAll varName='P' />  
580     <r:keyHolder>  
581       <r:info>  
582         <ds:KeyValue>FDFEWEFF...</ds:KeyValue>  
583       </r:info>
```

```

584     </r:keyHolder>
585     <r:issue/>
586     <r:grant>
587         <r:keyHolder varRef='K' />
588         <r:possessProperty/>
589         <r:propertyAbstract varRef='P' />
590     </r:grant>
591 </r:grant>
592 <r:issuer>
593     <ds:Signature>...</ds:Signature>
594 </r:issuer>
595 </r:license>

```

596 The following license is an example issuer authorization license for authorizing an issuer to issue
597 sender authorization licenses.

```

598 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
599     <r:grant>
600         <r:forAll varName='K' />
601         <r:forAll varName='R' />
602         <r:keyHolder>
603             <r:info>
604                 <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
605             </r:info>
606         </r:keyHolder>
607         <r:issue/>
608         <r:grant>
609             <r:keyHolder varRef='K' />
610             <sx:rightUri definition='...' />
611             <r:resource varRef='R' />
612         </r:grant>
613     </r:grant>
614     <r:issuer>
615         <ds:Signature>...</ds:Signature>
616     </r:issuer>
617 </r:license>

```

618 The following license is an example issuer authorization license for authorizing an issuer to issue
619 (to other issuers) issuer authorization licenses allowing those other issuers to issue simple
620 attribute licenses, such as those that can be used for authentication or confidentiality.

```

621 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
622     <r:grant>
623         <r:forAll varName='I' />
624         <r:keyHolder>
625             <r:info>
626                 <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
627             </r:info>
628         </r:keyHolder>
629         <r:issue/>
630         <r:grant>
631             <r:forAll varName='K' />
632             <r:forAll varName='P' />
633             <r:keyHolder varRef='I' />
634             <r:issue/>
635             <r:grant>
636                 <r:keyHolder varRef='K' />
637                 <r:possessProperty/>
638                 <r:propertyAbstract varRef='P' />
639             </r:grant>
640         </r:grant>
641     </r:grant>
642     <r:issuer>
643         <ds:Signature>...</ds:Signature>
644     </r:issuer>

```

645

</r:license>

646

647

5 Threat Model and Countermeasures (Informative)

648

649 This section addresses the potential threats that a SOAP message may encounter and the
650 countermeasures that may be taken to thwart such threats. A SOAP message containing licenses
651 may face threats in various contexts. This includes the cases where the message is in transit,
652 being routed through a number of intermediaries, or during the period when the message is in
653 storage.

654 The use of licenses with WS-Security introduces no new threats beyond those identified for the
655 REL or WS-Security with other types of security tokens. Message alteration and eavesdropping
656 can be addressed by using the integrity and confidentiality mechanisms described in WS-
657 Security. Replay attacks can be addressed by using of message timestamps and caching, as well
658 as other application-specific tracking mechanisms. For licenses, ownership is verified by the use
659 of keys; man-in-the-middle attacks are generally mitigated. It is strongly RECOMMENDED that all
660 relevant and immutable message data be signed. It should be noted that transport-level security
661 MAY be used to protect the message and the security token. In order to trust licenses, they
662 SHOULD be signed natively and/or using the mechanisms outlined in WS-Security. This allows
663 readers of the licenses to be certain that the licenses have not been forged or altered in any way.
664 It is strongly RECOMMENDED that the <r:license> elements be signed (either within the token,
665 as part of the message, or both).

666 The following few sections elaborate on the afore-mentioned threats and suggest
667 countermeasures.

668

5.1 Eavesdropping

669 Eavesdropping is a threat to the confidentiality of the message, and is common to all types of
670 network protocols. The routing of SOAP messages through intermediaries increases the potential
671 incidences of eavesdropping. Additional opportunities for eavesdropping exist when SOAP
672 messages are persisted.

673 To provide maximum protection from eavesdropping, licenses, license references, and sensitive
674 message content SHOULD be encrypted such that only the intended audiences can view their
675 content. This removes threats of eavesdropping in transit, but does not remove risks associated
676 with storage or poor handling by the receiver.

677 Transport-layer security MAY be used to protect the message from eavesdropping while in
678 transport, but message content must be encrypted above the transport if it is to be protected from
679 eavesdropping by intermediaries.

680

5.2 Replay

681 The reliance on authority protected (e.g. signed) licenses to <r:keyHolder> principals precludes
682 all but the key holder from binding the licenses to a SOAP message. Although this mechanism

683 effectively restricts message authorship to the holder of the confirmation key, it does not preclude
684 the capture and resubmission of the message by other parties.

685 Replay attacks can be addressed by using message timestamps and caching, as well as other
686 application-specific tracking mechanisms.

687 **5.3 Message Insertion**

688 This profile of WS-Security is not vulnerable to message insertion attacks. Higher-level protocols
689 built on top of SOAP and WS-Security should avoid introducing message insertion threats and
690 provide proper countermeasures for any they do introduce.

691 **5.4 Message Deletion**

692 This profile of WS-Security is not vulnerable to message deletion attacks other than denial of
693 service. Higher-level protocols built on top of SOAP and WS-Security should avoid introducing
694 message deletion threats and provide proper countermeasures for any they do introduce.

695 **5.5 Message Modification**

696 Message Modification poses a threat to the integrity of a message. The threat of message
697 modification can be thwarted by signing the relevant and immutable content by the key holder.
698 The receivers SHOULD only trust the integrity of those segments of the message that are signed
699 by the key holder.

700 To ensure that message receivers can have confidence that received licenses have not been
701 forged or altered since their issuance, licenses appearing in <wsse:Security> header elements
702 SHOULD be integrity protected (e.g. signed) by their issuing authority. It is strongly
703 RECOMMENDED that a message sender sign any <r:license> elements that it is confirming and
704 that are not signed by their issuing authority.

705 Transport-layer security MAY be used to protect the message and contained licenses and/or
706 license references from modification while in transport, but signatures are required to extend such
707 protection through intermediaries.

708 **5.6 Man-in-the-Middle**

709 This profile of WS-Security is not vulnerable to man-in-the-middle attacks. Higher-level protocols
710 built on top of SOAP and WS-Security should avoid introducing Man-in-the-Middle threats and
711 provide proper countermeasures for any they do introduce.

712

713

6 References

- 714 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels,"
715 RFC 2119, Harvard University, March 1997,
716 <http://www.ietf.org/rfc/rfc2119.txt>
- 717 **[REL]** ISO/IEC 21000-5:2004, "Information technology -- Multimedia framework
718 (MPEG-21) -- Part 5: Rights Expression Language,"
719 [http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUM
720 BER=36095&ICS1=35&ICS2=40&ICS3=](http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36095&ICS1=35&ICS2=40&ICS3=)
- 721 **[SOAP]** D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.
722 Frystyk Nielsen, S Thatte, D. Winer. Simple Object Access Protocol
723 (SOAP) 1.1, W3C Note 08 May 2000, <http://www.w3.org/TR/SOAP/>
724
725 W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging
726 Framework", 23 June 2003
- 727 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers
728 (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox
729 Corporation, August 1998, <http://www.ietf.org/rfc/rfc2396.txt>
730
731 T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers
732 (URI): Generic Syntax," RFC 3986, MIT/LCS, Day Software, Adobe
733 Systems, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>.
- 734 **[WS-Security]** OASIS Standard 200401, "Web Services Security: Soap Message
735 Security 1.0 (WS-Security 2004)," March 2004, [http://docs.oasis-
736 open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
737
738 OASIS Standard, "Web Services Security: Soap Message Security 1.1
739 (WS-Security 2005)," TBD 2005, [http://docs.oasis-
740 open.org/wss/2005/xx/oasis-2005xx-wss-soap-message-security-1.1.pdf](http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-soap-message-security-1.1.pdf)
- 741 **[XML-ns]** T. Bray, D. Hollander, A. Layman. Namespaces in XML. W3C
742 Recommendation. January 1999, [http://www.w3.org/TR/1999/REC-xml-
743 names-19990114](http://www.w3.org/TR/1999/REC-xml-names-19990114)
- 744 **[XML Signature]** D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. XML-
745 Signature Syntax and Processing, W3C Recommendation, 12 February
746 2002, <http://www.w3.org/TR/xmlsig-core/>
747

Appendix A: Acknowledgements

Contributors:

Maneesh	Sahu	Actional Corporation
Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Laboratory
Hal	Lockhart	BEA Systems, Inc.
Corinna	Witt	BEA Systems, Inc.
Steve	Anderson	BMC
Richard	Levinson	Computer Associates
Davanum	Srinivas	Computer Associates
Thomas	DeMartini	ContentGuard, Inc.
Guillermo	Lao	ContentGuard, Inc.
TJ	Pannu	ContentGuard, Inc.
Xin	Wang	ContentGuard, Inc.
Merlin	Hughes	Cybertrust
Sam	Wei	Documentum
Tim	Moses	Entrust
Carolina	Canales-Valenzuela	Ericsson
Dana	Kaufman	Forum Systems, Inc.
Toshihiro	Nishimura	Fujitsu Limited
Kefeng	Chen	GeoTrust
Irving	Reid	Hewlett-Packard
Kojiro	Nakayama	Hitachi
Paula	Austel	IBM
Derek	Fu	IBM
Maryann	Hondo	IBM
Kelvin	Lawrence	IBM
Hiroshi	Maruyama	IBM
Michael	McIntosh	IBM
Anthony	Nadalin	IBM
Nataraj	Nagaratnam	IBM
Ron	Williams	IBM
Don	Flinn	Individual
Bob	Morgan	Internet2
Kate	Cherry	Lockheed Martin
Paul	Cotton	Microsoft Corporation
Vijay	Gajjala	Microsoft Corporation
Alan	Geller	Microsoft Corporation
Martin	Gudgin	Microsoft Corporation
Chris	Kaler	Microsoft Corporation
John	Shewchuk	Microsoft Corporation
Jeff	Hodges	Neustar, Inc.
Frederick	Hirsch	Nokia
Senthil	Sengodan	Nokia
Abbie	Barbir	Nortel Networks
Lloyd	Burch	Novell
Charles	Knouse	Oblix
Vamsi	Motukuru	Oracle

Ramana	Turlapati	Oracle
Prateek	Mishra	Principal Identity
Andrew	Nash	Reactivity
Ben	Hammond	RSA Security
Rob	Philpott	RSA Security
Martijn	de Boer	SAP
Blake	Dournaee	Sarvega
Coumara	Radja	Sarvega
Pete	Wenzel	SeeBeyond Technology Corporation
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet
Symon	Chang	Tibco Software Inc.
John	Weiland	US Dept of the Navy
Hans	Granqvist	VeriSign
Phillip	Hallam-Baker	VeriSign
Hemma	Prafullchandra	Verisign

750

751

Appendix B: Revision History

Rev	Date	What
01	27-May-2005	Initial draft based on REL Token Profile 1.0, updated for 1.1.
02	14-Jun-2005	Marked as Committee Draft.

752

753